Fearless Refactoring

Kevin Cottrill
@KevinCottrill88

David Berry
@DavidCBerry13

https://github.com/CleanCodeTechTalks/FearlessRefactoring/

Refactoring

Changing the internal design and structure of code without changing its behavior so the code is easier to maintain and understand.

Why write automated tests?

Test Your Software, or Your Users Will

Test ruthlessly. Don’t make your users find bugs for you

Find Bugs Once

Once a human tester finds a bug, it should be the last time a human tester finds that bug. Automatic tests should check for it from then on.

The Pragmatic Programmer

Dependency Injection

Instead of a class directly creating a dependency by calling new() on the dependent object, the dependent object is passed in (injected) into the class

This allows the dependent object to be easily swapped out for different object that implements the same interface

Traditional Dependencies

```
public class LocationController : Controller
{


    private FoodTruckContext dbContext;
    private GeocoderClient geocoderClient;


    public LocationController()
    {
        dbContext = new FoodTruckContext();
        geocoderClient = new GeocoderClient();
    }
}
```

- Component creates dependencies directly
- To change these dependencies, we have to edit the code
- This makes it hard to replace external components with stand in components for testing

Using Dependency Injection

```
public class LocationController : Controller
{

    private FoodTruckContext dbContext;
    private IGeocoderClient geocoderClient;


    public LocationController(FoodTruckContext context,
        IGeocoderClient geocoder)
    {
        dbContext = context;
        geocoderClient = geocoder;
    }
}
```

- Dependencies are passed in through the constructor
- We can now easily pass in mock objects when we test this object

Mock Object

A stand in object we use during testing to return the exact data we want for a certain test

This way we can isolate our code from external dependencies and simulate those dependencies returning the exact data we need for our test conditions

Unit Testing Tools

xUnit.net

https://xunit.github.io/

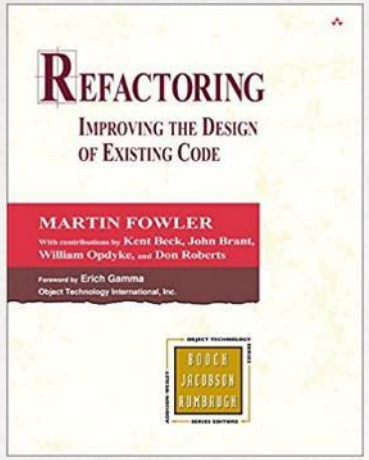
https://github.com/moq/moq4

IOC Containers

https://autofac.org/

http://www.ninject.org/

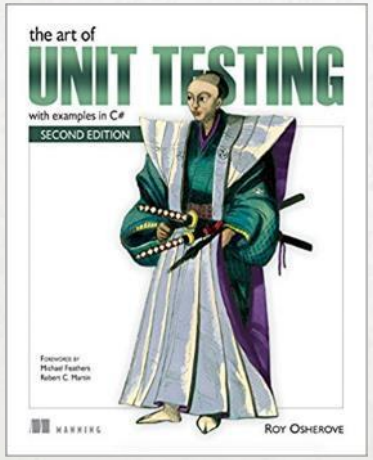
Resources



Refactoring: Improving the Design of Existing Code

Martin Fowler

https://www.amazon.com/Refactoring-Improving-Design-Existing-Code/dp/0201485672



The Art of Unit Testing

Roy Osherove

https://www.amazon.com/Art-Unit-Testing-examples/dp/1617290890

Catalog of Refactorings

https://refactoring.com/catalog/

Using the Catalog ▶

Tags

- associations
- encapsulation
- generic types
- interfaces
- class extraction
- GOF Patterns
- local variables
- vendor libraries
- errors
- type codes
- method calls
- organizing data
- inheritance
- conditionals
- moving features
- composing methods
- defining methods

- Add Parameter
- Change Bidirectional Association to Unidirectional
- Change Reference to Value
- Change Unidirectional Association to Bidirectional
- Change Value to Reference
- Collapse Hierarchy
- Consolidate Conditional Expression
- Consolidate Duplicate Conditional Fragments
- Decompose Conditional
- Duplicate Observed Data
- Dynamic Method Definition
- Eagerly Initialized Attribute
- Pull Up Constructor Body
- Pull Up Field
- Pull Up Method
- Push Down Field
- Push Down Method
- Recompose Conditional
- Remove Assignments to Parameters
- Remove Control Flag
- Remove Middle Man
- Remove Named Parameter
- Remove Parameter
- Remove Setting Method
- Remove Unused Default Parameter

Thank You

Source Code and Slides

https://github.com/CleanCodeTechTalks/FearlessRefactoring/

Contact Info

Kevin Cottrill - @KevinCottrill88
David Berry - @DavidCBerry13